⑦ Loops $\quad i \quad J \quad K$

$\quad n/2 \quad \log n \quad \log n$

$$TC = n/2 * \log n * \log n$$

$$= O(n(\log n^2 n)^2)$$

⑧ Orderloop $\quad i \quad J$

$\quad n \quad m$

$n/3$

$$TC = O(n^3)$$

⑨ $i \quad j$

| | |
|---|---|
| 1 | n times |
| 2 | n/2 times |
| 3 | n/2 times |
| ⋮ | ⋮ |
| n | n/2 times. |

$$TC = O(n \log n)$$

⑩ Since polynomial grow slower than exponential $n^k$ has an asymptotic upperbound of $O(a^n)$ for $a=2$, $K0=2$

④ $T(n) = \{2T(n-1)-1$ if $n>0$, otherwise $1\}$

$T(n-1) = 2[2T(n-2)-1]-1$

$= 2^2(T(n-2))-2-1$

$T(n-2) = 2(2^2(T(n-3)-1))-2-1$

$T(n-2) = 2^3 T(n-3)-4-2-1$

$T(n-3) = 2(2^3(T(n-4)-1))-4-2-1$

$= 2^4(T(n-4))-8-4-2-1$

$\vdots$

$= 2^n(T(n-n))-2^{n-1}-2^{n-2}-\cdots-2^0$

$T(0) = 1$

$2^n-2^{n-1}-2^{n-2}-\cdots-2^0$

$\Rightarrow 2^n-(2^n-1)$

$TC = O(1)$

⑤ $S = 1, 3, 6, 10 \cdots n$

$\frac{K(K+1)}{2} = n$

$K^2 = n$

$K = \sqrt{n}$

$TC = O(\sqrt{n})$

⑥ $TC = O(\sqrt{n})$

$i = 1, 2, 4, 8 \ldots\ldots n$ $\quad a = 1, r = 2$

$t_K = a r^{K-1}$

$n = \frac{2^K}{r} \Rightarrow 2n = 2^K$

taking log on both side

$K \log_2 2 = \log_2 (n) + \log_2 (2)$

$K = \log_2 (n) + 1$

$O(\log_2 (n) + 1)$

$O(\log n)$

③ $T(n) = \{ 3T(n-1)$ if $n > 0$ otherwise $1 \}$

Using backward substitution

$T(n-1) = 3[3T(n-2)]$

$T(n-1) = 3^2[T(n-2)]$

$T(n-1) = 3^2[3T(n-2+1)]$

$= 3^3 T(n-1)$

$= \vdots$

$= 3^n(T(n-n))$

$= 3^n(T(0))$

$T(0) = 1$

$TC = O(3^n)$

DAA Tutorial - I

NAME → TARUN KUMAR DHIMAN
ROLL NO → 1918917
Sec → IT

① Asymptotic Notation :- There are notation or expression that are used to respresent the Complexity of an algorithm.

Types

① Theta $\Theta$ → Curve the Legend in which the function will fluctuate.

② Big [O] → $f(n) = O(g)(n)$

g(n) is "High" upper bound of f(n)
i.e f(n) can never go beyond g(n)

③ Omega ($\Omega$) $f(n) = \Omega\, g(n)$

g(n) is "High" Lower bound of f(n)
i.e f(n) will never perform better than g(n)

② Time Complexity of

for (i=0 to n) { i=i*2;}